Homework 1: Review of Proof Techniques

Due: January 28, 2026 at 11:59 p.m.

This homework must be typed in LaTeX. Please use the code released as part of the homework, remove these instructions and fill in the solutions.

Please ensure that your solutions are complete, concise, and communicated clearly. Use full sentences and plan your presentation before you write. Except where indicated, consider every problem as asking for a proof.

Your solutions must be submitted using Gradescope. When doing so, please correctly mark the pages with the solutions to each problem.

Problem 1. Let $f : A \to B$ and $g : B \to C$ be functions. Prove or disprove the following statement:

If $g \circ f$ is bijective, then f is injective and g is surjective.

Solution. □

Problem 2. Prove or disprove the following statement:

Prove that the square for every odd number can be expressed as $8k + 1$ *for some integer* k

Solution. □

Problem 3. You are given a rectangular chocolate bar with $m \times n$ squares of chocolate, and our task is to divide it into mn individual squares. you are only allowed to split one piece of chocolate at a time using a vertical or a horizontal break. For example, suppose that the chocolate bar is 2×2. The first split makes two pieces, both 2×1. Each of these pieces requires one more split to form single squares. This gives a total of three splits. Use an induction argument to prove the correctness of the following statement:

$mn - 1$ splits are sufficient to divide a rectangular chocolate bar with mn squares into individual squares.

Solution. □

Problem 4. Consider the algorithm given as pseudocode below:

1. What is the output of the algorithm? Provide an informal but precise description.
2. Prove the correctness of the algorithm.
3. Analyze the running time of the algorithm.

Algorithm 1 ?-?

```
Input: An $n$-vertex graph represented by an adjacency matrix $\mathbf{D}$, where $\mathbf{D}[i][j]$ is
the non-negative weight of the edge from vertex $i$ to vertex $j$, for $0 \leq i, j < n$.
If there is no edge connecting $i$ and $j$, $\mathbf{D}[i][j] = \infty$.
Output: ?
for $i \leftarrow 0$ to $n-1$ do                                    ▷ Initialization solution
    for $j \leftarrow 0$ to $n-1$ do
        $\mathbf{S}[i][j] \leftarrow \mathbf{D}[i][j]$
for $k \leftarrow 0$ to $n-1$ do
    for $i \leftarrow 0$ to $n-1$ do
        for $j \leftarrow 0$ to $n-1$ do
            $\mathbf{S}[i][j] \leftarrow \min\{\mathbf{S}[i][j], \mathbf{S}[i][k] + \mathbf{S}[k][j]\}$
```

Solution. □

Problem 5. A *bit*, as you probably know, is a value that is either 0 or 1. A nonnegative integer can be written in binary, which means base two, and the binary representation can be interpreted as a sequence of bits. For example, the number thirteen is represented in binary as 1011. The rightmost position is the one's position, the second-to-rightmost position is the two's position, the third-to-rightmost position is the four's position, and so on.
A *bitwise operator* operates on nonnegative integers by interpreting the integers as sequences of bits.

- AND (&): The AND of two nonnegative integers is the integer whose binary representation obeys the following: for each bit position, the output bit is the AND of the two input bits in the same position.
  Example: `1100 & 1010 = 1000`
- OR (|): The OR of two nonnegative integers is the integer whose binary representation obeys the following: for each bit position, the output bit is the OR of the two input bits in the same position.
  Example: `1100 |  1010 = 1110`
- Exclusive OR (aka XOR, $\oplus$, ^): The XOR of two nonnegative integers is the integer whose binary representation obeys the following: for each bit position, the output bit is the XOR of the two input bits in the same position.
  Example: `1100 ^ 1010 = 0110`

Bitwise AND, OR, and XOR are commutative and associative.

Here are two more operators that interpret values as binary:

- Bit right shift ($\gg$): The left operand value, interpreted as a sequence of bits, is shifted right by the number n of bits specified by the right operand value. Because the output is required to be an integer, the n lowest-order bit are discarded.
  Example: `11010 >> 2 = 110`
- Bit left shift ($\ll$): The left operand value, interpreted as a sequence of bits, is shifted left by the number n of bits specified by the right operand value. Zeros are inserted and become the rightmost n bits.
  Example: `1011 << 2 = 101100`

1. Evaluate the following expressions:
   (i) $1000 \oplus 111 \oplus 11 \oplus 1$
   (ii) $(111111 \gg 5) \ll 5$
   (iii) $1001 \& 1011 | 1000$
2. What are the identity elements of bitwise AND, OR, and XOR?
3. Identify the bitwise operation `x ? y` equivalent to the following operations in Python:
   (i) `x * (2 ** y)`
   (ii) `x // (2 ** y)`
4. Say we have a large positive integer x. How can we find $x \bmod 2^4$ without using mod (instead using a binary operation)?

Solution. □